Semantic Wiki Search

Peter Haase[1], Daniel Herzig[1,⋆], Mark Musen[2], and Thanh Tran[1]

[1] Institute AIFB, Universität Karlsruhe (TH), Germany
{pha,dahe,dtr}@aifb.uni-karlsruhe.de
[2] Stanford Center for Biomedical Informatics Research (BMIR)
Stanford University, USA
musen@stanford.edu

Abstract. Semantic wikis extend wiki platforms with the ability to represent structured information in a machine-processable way. On top of the structured information in the wiki, novel ways to search, browse, and present the wiki content become possible. However, while powerful query languages offer new opportunities for semantic search, the syntax of formal query languages is not adequate for end users. In this work we present an approach to semantic search that combines the expressiveness and capabilities of structured queries with the simplicity of keyword interfaces and faceted search. Users articulate their information need in keywords, which are translated into structured, conjunctive queries. This translation may result in multiple possible interpretations of the information need, which can then be selected and further refined by the user via facets. We have implemented this approach to semantic search as an extension to Semantic MediaWiki. The results of a user study in the SMW-based community portal *semanticweb.org* show the efficiency and effectiveness of the approach as well as its ease of use.

1 Introduction

The availability of structured information on the Semantic Web enables new opportunities for information access. Search is no longer limited to matching keywords against documents, but instead complex information needs can be expressed in a structured way, with precise and structured answers as results [1,2,3].

A prominent application area are semantic wikis. Wikis in general have become a popular tool for collaborative creation and exchange of content on the Web. The content typically is maintained in wiki pages with plain text and occasional markup elements. Semantic wikis aim at extending wiki platforms with the ability to represent structured information in a machine-processable way. For example, Semantic MediaWiki (SMW) [4] is an extension of the well-known MediaWiki, which also powers Wikipedia. The main element of this extension is the notion of Typed Links, which enable users to generate structured information with a well-defined semantics. On top of the structured information in the wiki, novel ways to reuse, browse, and search the wiki content become possible.

For example, SMW already supports the so called ASK language[1] to perform structured queries. However, while such powerful query languages offer new opportunities

⋆ Work done while visiting student at BMIR, Stanford University.

[1] http://semantic-mediawiki.org/wiki/Help:Semantic_search

L. Aroyo et al. (Eds.): ESWC 2009, LNCS 5554, pp. 445–460, 2009.

for semantic search, users typically are not willing to use the syntax of formal query languages. What would be desired is a search interface that combines the expressiveness and capabilities of structured queries with the simplicity known from keyword interfaces and faceted browsing, which are much easier to handle for lay end users.

In this work we present such an approach to semantic search: Users articulate their information need using keyword queries, which are translated by the system into structured queries representing possible interpretations of the information need. The user is able to select the query that best matches his information need and to further refine it via facets. In doing so, the user does not need any knowledge about formal query languages nor any schema of the data. We extend our previous work [5] on semantic search based on a translation of keyword-based queries into structured queries against graph-structured data, thus enabling expressive queries while retaining a simple end user interface. We apply this approach to semantic search in the Semantic MediaWiki. We have deployed our extension for semantic search on several installations of Semantic MediaWiki. One of them is the wiki-based portal of the Semantic Web community – *semanticweb.org*, which gathers (semi-)structured information about news, people, events, etc. relevant for the Semantic Web community. Within *semanticweb.org* we have performed a user study that shows the efficiency and effectiveness of our approach.

The rest of the paper is organized as follows. In Section 2 we present on overview of our approach to semantic search from a process perspective. We describe the technical details of this process as well as the implementation in Section 3. We report on the results of our user evaluation study in Section 4. After a discussion of related work in Section 5 we conclude in Section 6.

2 Semantic Search – A Process View

In this section we describe the search process underlying our approach to semantic search. This process comprises three main steps:

1. *Articulation of information need:* Following the paradigm of keyword search, users express their information need using keyword queries.
2. *Query interpretation using keyword translation:* The information need expressed in keywords is interpreted via a translation into conjunctive queries. The user selects the query that best matches his information need.
3. *Result presentation and query refinement:* The answers to the selected query are presented to the user. The user can further refine the query using faceted search.

This process is based on a paradigm that differs from traditional search, where users issue a query, obtain a (set of) results, and – if the results do not fulfill the information need – start over with issuing a new query.

First, we introduce an additional step, in which different possible interpretations of the user information need are computed and presented to the user. Instead of presenting the results directly, which might actually belong to many distinct queries (representing different information needs), we allow the user to select the correct interpretation.

Second, we allow the user to refine the interpreted query to match the exact information need. The act of refinement is thus much more precise and direct: Instead of having to issue a new query, the user can directly refine the possible interpretations.

In the following, we briefly discuss some important aspects of these individual steps.

Articulation of the information need. In our approach, users articulate their information needs using keyword queries. We believe this is the most adequate form, as keyword interfaces have been widely adopted, and users are familiar with them both due to their simplicity and their presence in today's systems. For searching, the users do not need to know about the query syntax, the schema and even the labels of the data elements. Ideally, they can use their own words to express their information needs.

Query interpretation using keyword translation. This step is concerned with the translation of the user queries into system queries, i.e. structured conjunctive queries. Here we interpret keywords as elements of structured queries. The keyword search process contains an additional step, namely the presentation of structured queries. We consider this step as beneficial because these queries can be seen as descriptions, and can thus facilitate the comprehension of the answers. Also, refinement can be made more precisely on the structured query than on the keyword query. The details of the keyword translation will be described in Section 3.2. Briefly summarized, we follow a graph exploration to identify query graphs, i.e. substructures that connect all keyword elements. According to a query ranking scheme, the computed query graphs are sorted and presented to the user for *selection*. Here, the queries are not presented using a formal syntax, but using an intuitive, graph-based representation. Additionally, to better understand the query, we also present snippets of the query results. In certain cases, these snippets may already be the answers that fulfill the information need and the final step may not be needed.

Result presentation and refinement. Query results – which in the general case are sets of tuples satisfying the conjunctive query – are presented to the user in a structured, tabular form. Further, the user can refine the query following the paradigm of faceted search. Faceted search (often also referred to as faceted browsing) is a paradigm for exploring data through navigating along its hierarchical or multidimensional metadata or structure. The main characteristic of this paradigm is that it allows users to narrow and expand the data of interest according to their information need in an interactive way. The user actions of adding or removing facets are transparently converted to operations on the conjunctive query. The refined queries are immediately evaluated and the new results presented without the user having to explicitly issue a new query.

Refinement may be needed for several reasons. First of all, the computed interpretations may not exactly match the information need. Second, the user may also start out with an ill-defined information need, not knowing what exactly he is searching for. For these cases, the system can be used for faceted browsing to explore the knowledge base. The details of the faceted search will be described in Section 3.3.

3 Semantic Search in SMW

In this section we describe how the process of semantic search is realized and supported in our extension to Semantic MediaWiki. We first present the underlying data and query model within SMW, and then discuss the aspects of query interpretation, result presentation and query refinement.

3.1 Data and Query Model

Our search operates on the structured part of the data in Semantic MediaWiki. We briefly discuss the main primitives of structuring data in SMW along with a formal semantic interpretation of the wiki's structure in terms of OWL DL. The primary structural mechanism of MediaWiki is the organization of content within wiki pages. Every page can be assigned to one or many categories, which can be organised hierarchically. SMW introduces ways of adding further structure to MediaWiki by means of *annotating* textual content of the wiki: Properties are used to express binary relationships between one entity (as represented by a wiki page) and some other such entity or data value. The formal semantics of structured data in SMW is given via a mapping to the OWL ontology language [4].

Most elements are represented in OWL in a straight-forward manner, using the obvious mapping from wiki pages to OWL entities: normal article pages correspond to individuals, categories correspond to OWL classes, and SMW properties correspond to OWL properties. More precisely, SMW properties with article pages in their range are mapped to object properties, and SMW properties with datatypes in their range are mapped to datatype properties. Accordingly, property values can be individuals or data values (typed literals). Most annotations thus are directly mapped to simple assertions in OWL, similar to RDF triples. Finally, containment of pages in MediaWiki's categories is represented as class membership in OWL. MediaWiki supports the hierarchical organization of categories, and SMW can be configured to interpret this as an OWL class hierarchy. Moreover, SMW introduces a special property "subproperty of" that can be used for property hierarchies.

For querying, we distinguish between the notion of a *user query* and a *system query*. The user expresses his information needs in keyword queries.

Example 1. As a running example, we consider an information need of a user who is searching for upcoming conferences with their paper deadline, and as he likes to go to Greece, he is in particular interested in conferences in this country. He might express his information need using the keywords: "conferences Greece deadline".

The system queries are *conjunctive queries*. Formally, a *conjunctive query* is an expression of the form $(x_1, \ldots, x_k).\exists x_{k+1}, \ldots x_m.A_1 \wedge \ldots \wedge A_r$, where $x_1, \ldots, x_k$ are called *distinguished variables*, $x_{k+1}, \ldots, x_m$ are *undistinguished variables* and $A_1, \ldots, A_r$ are *query atoms*. These atoms are of the form $P(v_1, v_2)$, where P is called *predicate*, v_1, v_2 are variables or, otherwise, are called *constants*.

Conjunctive queries have high practical relevance because they are capable of expressing the large class of relational queries. The vast majority of query languages for many data models used in practice fall into this fragment, including large parts of SPARQL. Conjunctive queries can be used to address typical information needs frequently occurring when searching in a semantic wiki. We can distinguish the following categories of queries (corresponding to special cases of conjunctive queries):

1. *Entity Queries* are queries for certain entities (individuals in OWL), where the results correspond to a wiki page. An example information might be searching for the page of a person, e.g. "Rudi Studer".

2. *Fact Queries* are queries for concrete properties of particular objects (Example: "Rudi Studer phone"), however, the result does not correspond directly to a page, but to one (or more) statements on a page.
3. *General Conjunctive Queries* are queries that allow to retrieve n-ary tuple sets as results. An example could be the query "members AIFB phone" which asks for all members of the AIFB institute along with their phone numbers.[2]

In terms of the results, entity queries are similar to the existing keyword search supported in MediaWiki, which always returns single pages. The latter two types of queries go beyond the current search capabilities of MediaWiki.

3.2 Query Interpretation and Processing

The goal of this step is to interpret the keywords of the user query in terms of a structured, conjunctive query. Typically, due to the ambiguity inherent in keyword queries, such an interpretation is not unique. Therefore, we rely on a top-k procedure to generate candidate interpretation and allow the user to select the interpretation that best matches his information need. We build on our previous work [5] on translating keyword queries into structured queries based on a graph-exploration technique. For this purpose, we consider the knowledge base as a graph structure. (Intuitively, this graph corresponds to the RDF representation of an OWL knowledge base).

The computation of query graphs as interpretations of the user keywords involves three tasks: 1) construction of the query search space 2) top-k query graph exploration and 3) query graph ranking. Specific concepts and algorithms for these tasks have been introduced in [5].

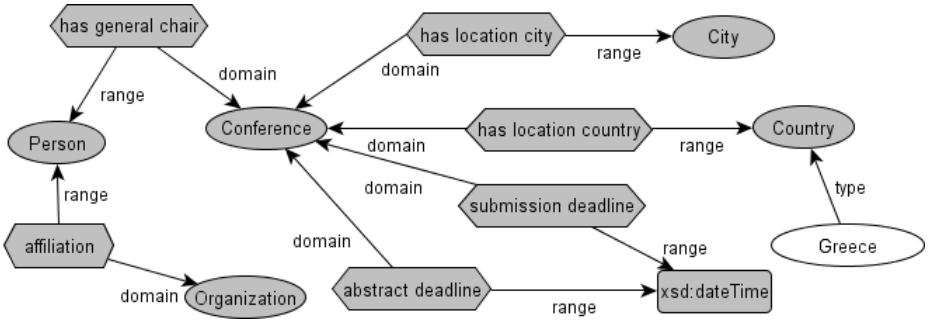

Fig. 1. Query space: schema graph (grey) augmented with keyword-matching elements (white)

Construction of the Query Search Space. The query search space contains all the elements that are necessary for the computation of possible interpretations. In our previous work [5], we have shown that keyword search is most efficient when the exploration for possible interpretations is performed on an "augmented" schema graph, instead of using the entire data graph (c.f. [6,7]). The schema graph can be trivially obtained from the

[2] Please note that this particular query could be represented using a tree-shaped conjunctive query, but in general the queries do not need to be tree-shaped.

class and property definitions (corresponding to the category and property definitions in the wiki). From experience we know that in semantic wikis the schema is typically incomplete. For example, often the domains and ranges of properties are not fully specified. Therefore, we additionally apply techniques for computing schema graphs automatically. In particular, a schema graph is derived from the data using the aggregation rules as described in [5].

The schema graph is augmented with elements that match the user keywords (to explore the query constants). We rely on a *keyword index* to map keywords to elements of the knowledge base, which in our approach might be classes, properties, individuals, and data values. IR concepts are adopted to support an imprecise matching that incorporates syntactic and semantic similarities. As a result, the user does not need to know the exact labels of the data elements when doing keyword search. Each element finally returned is associated with *a score* measuring the degree of matching, which is later used for ranking possible interpretations. The schema graph and the keyword elements are combined to obtain the query space.

Example 2. Figure 1 illustrates the query space constructed for our example keyword query. It consists of (the fragment of) a schema graph (nodes in grey). Keyword elements not covered by the schema graph are added. In particular, the keyword matching element *Greece* is added to *Country*, which it instantiates.

Exploration of Top-K Query Graphs. Given the query space, the remaining task is to search for the minimal query graphs in this space. Informally, a query graph is a matching subgraph of the augmented schema graph, such that for every keyword of the user query it contains at least one representative keyword matching element, and (2) the graph is connected, i.e. there exists a path from every graph element to every other graph element. A matching query graph is minimal if there exists no other query graph with a lower score. In [5], we have proposed a top-k procedure to find such query graphs. This procedure starts from the keyword elements and iteratively explores the query space for all distinct paths beginning from these elements. During this procedure, the path with the highest score so far is selected for further exploration. At some point, an element might be discovered to be a connecting element, i.e. there is a path from that element to at least one keyword element, for every keyword in the user query. These paths are merged to form a query graph. The explored graphs are added to the candidate list. The process continues until the upper bound score for the query graphs yet to be explored is lower than the score of the k-ranked query graph in the candidate list.

Example 3. An example query graph that can be found through exploration along these paths is shown in Figure 2. An alternative interpretation could have resulted in a query graph with *submission deadline* instead of *abstract deadline* as connecting element.

Scoring Query Graphs. During top-k computation, elements with highest scores are chosen for further exploration. Thus, the quality of the computed top-k query graphs depends largely on the scoring function. Factors that are considered in the element scoring function include the matching score (as obtained from the keyword index for the matching elements) and the importance (computed offline for elements of every schema graph). Since the cost of a path monotonically increases with the number of

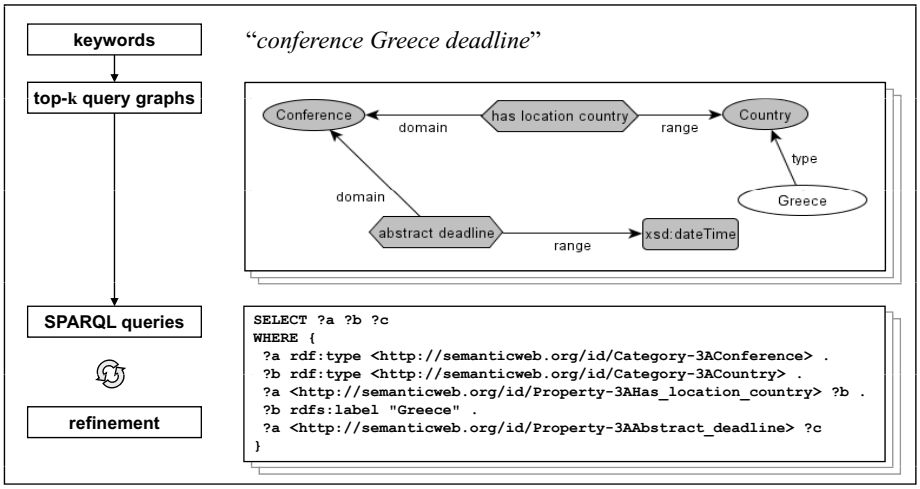

Fig. 2. Keyword translation

constituent elements, the length is also implicitly captured by this cost function. For the details of the scoring function, we refer to [5].

Query translation results in a list of top-k query graphs. Finally, the query graphs are translated to conjunctive queries in SPARQL query syntax. (See bottom of Figure 2 for an example.) Basically, edges are mapped to predicates, whereas vertices are mapped to variables and constants of the conjunctive query. (See [5] for a detailed description.)

3.3 Result Presentation and Query Refinement by Faceted Search

Since our search aims at lay end users, the interpretations in the form of conjunctive queries have to be presented to the user in an comprehensible and intuitively understandable way, so that the user can select the interpretation that fits best to his information need. A conjunctive query in SPARQL itself is far from being intuitively understandable, especially for end users. Therefore, we developed a concept for visualizing conjunctive queries suited for wiki-environments. We choose a table layout as the major pattern with a nested but minimal graph structure. The layout table is divided horizontally into three sections from top to bottom: A head-row, a relation-row and a preview section. The relation row spans the entire table, whereas the head row and the preview section are vertically divided into columns. There is one column for each distinguished variable of the query. The column labels denote the entity types (classes) and the arrows in between them represent the relationships (properties). In case there is no type information available, either the name of a property having the corresponding variable as the range is displayed in the column head, or if it occurs only as the domain of a property the placeholder "Something" is shown. All entities and relations tool have tips, which show the wiki page of the entity to help the user understand what entity is shown. To help the user grasp the meaning of the interpretation, the preview section contains a snippet of the query's results. The result tuples shown here are bindings to distinguished variables and are individuals of the classes denoted by the respective columns or data values.

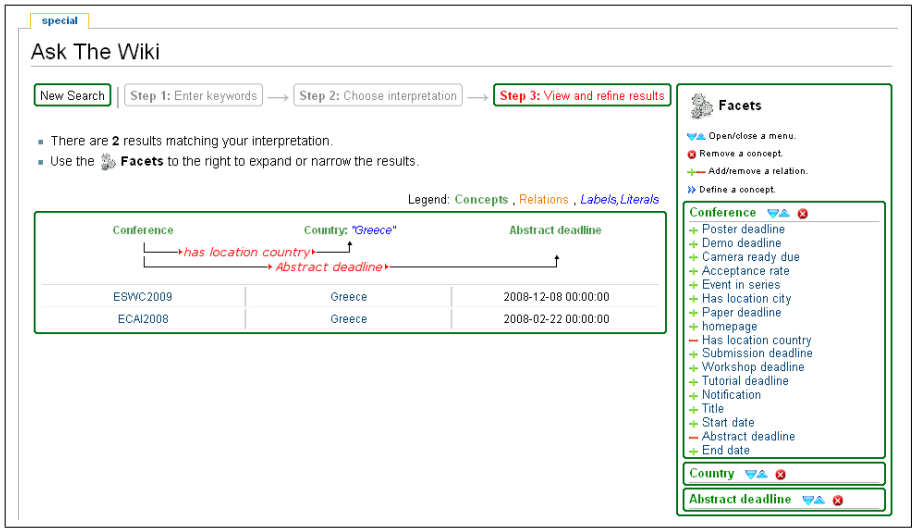

Fig. 3. Result presentation and query refinement. The selected query along with the query results are shown in the center. On the right side is the menu providing the faceted search capabilities.

This concept is sufficiently general for presenting answers conjunctive queries, in particular the three mentioned types of queries (cf. section 3.1).

1. *Entity Queries:* One single column for the entities in question.
2. *Fact Queries:* Two columns, one for a defined class and one for a data value/class.
3. *General Conjunctive Queries:* Several columns and relationships between them.

Figure 3 shows the visualization of the query from the example in Figure 2.

When an interpretation is chosen, the preview section is expanded and shows all matching entity tuples. In addition, the *facets menu* is loaded on the right side of the screen (Figure 3). The facets menu offers the faceted search capabilities. The facets menu has a sub menu for each distinguished variable of the query presenting the possible facets. Facets are properties or attributes of entities. The query modifications are performed by adding or removing facets to the interpretation. For example, in Figure 3 the user could drill down and refined the search result by adding the property "submission deadline" to (class) "Conference" in the interpretation. The new interpretation is immediately evaluated and the result presented to the user. We present only properties to the user, which would lead to a non-empty result when added to the interpretation. In case the range concept of an object property is not defined, a sub menu in the same style would offer the user to choose one from a list of valid concepts, which would narrow the search result accordingly. Instead of a adding a property, the user could also refine the search result by removing the property "has located country", he would then get all conferences and their abstract deadlines without the constraint that they are located in Greece. Furthermore, the user could also remove a variable and all properties bound to it, by pressing the "x" button in the head line of the corresponding sub menu. Using

these modifications the user can fluently *browse* through the structured data of the semantic wiki by adapting the SPARQL query to his information need. All refinements are operations, namely adding or removing triple patterns to/from the SPARQL query.

3.4 Implementation

Our implementation – called *Ask The Wiki* – consists of two major parts, a Semantic MediaWiki (SMW) extension and a back-end, which we describe in the following. *Ask The Wiki* is integrated into the Semantic MediaWiki as a "Special page extension". This extension provides the user interface and the faceted search methods via SPARQL modification. It is implemented in PHP and uses AJAX for the user interaction. Although, we precompute the schema, because it is often not fully specified in the wiki, the facets are retrieved on the fly for each query individually via AJAX from the back-end. All user selections, query modifications, and process states are kept on the client side.

The back-end is realized as a Java Servlet running on a Tomcat server. The servlet provides the keyword translation and top-k query construction, as well as the query evaluation. Before the keyword translation and query construction are available online, an offline preprocessing step is required. This step comprises computing the schema graph and indexing the OWL data export of SMW using special schema and keyword indexes, which are realized using Lucene. For storing the data and processing the SPARQL queries, in principle any triple store exposing a SPARQL endpoint can be used. We use Sesame with its native store and enabled inferencing.

The implementation is compatible with SMW 1.2 (and above) and Tomcat 5.5 (and above) running on Java 6. We have already deployed the system on several Semantic MediaWikis. In the following section, we report on evaluation experiments in one of the installations of *Ask the Wiki*.

4 Evaluation

The goal of the evaluation was to assess the potential and ability of our approach to semantic search in a real-life application. Since the search should make the potential of the underlying semantic technologies available to end users, we performed a user study to evaluate the system in terms of effectiveness and efficiency, as well as user satisfaction and usability.

4.1 Data Set and Evaluation Setting

We performed the evaluation within the community portal *semanticweb.org*, a wiki-based platform serving the Semantic Web community. The wiki contains information about the Semantic Web, in particular (but not limited to) events, publications, tools, and people. Its OWL data is available in the RDF/XML format and comprises a total of 55,365 triples, with 657 classes, 948 properties, 27,778 property instances, and 11275 individuals (as of Dec 4 2008). The data has been created by the users of the wiki over the last three years. Since the nature of a wiki is to provide unconstrained user editing, the data does not follow a predefined vocabulary or strict schema. The dataset is available at `http://semanticweb.org/RDF`. The search interface is publicly accessible at `http://semanticweb.org/wiki/Special:ATWSpecialSearch`.

The participants of the user study were 14 volunteers from the four different organizations active in the Semantic Web community. We chose a task based user evaluation with each task representing an information need that could typically occur when using the portal. Afterwards we asked the participants to answer a multiple choice questionnaire about their experience. The questions concerned their technical background (experience with other search engines, familiarity with certain technologies, etc.) and the experience and satisfaction with certain aspects of our search process. Additionally, the participants could give free text comments.

Each participant got five tasks and had up to three minutes to solve each task. The participants could give up before, if they felt that they could not solve the task. The participant received very limited information about the search interface upfront, namely that the search consists of three steps and that the search will not return a list of links like the common web search engines, but an interpretation of their keywords, also that they have to choose an interpretation, if necessary and that they could modify the interpretation in the third step. However, there was no walk-through introduction or information how to perform these steps.

The tasks were constructed so that all aspects of the systems functionality were covered, with different levels of difficulty. In particular, we created tasks that required queries that fall into the categories introduced before (i.e. entity queries such as *Find the page describing the AIFB institute*, fact queries such as *When is the paper deadline for the ASWC2008*, general conjunctive queries such as *Find the capitals of countries in Europe and the population of these cities*). Actions taken by the participants and system responses were logged. In particular, we logged the users' steps and keyword inputs and the system responses and measured how often users could solve the task, how much time it took them to choose an interpretation and how well the system performed in terms of keyword translation, query construction, and query evaluation. The evaluation was performed based on both the analysis of the log files as well as the questionnaire.

It should be noted that standard IR evaluation measures such as precision and recall do not directly apply to our approach to search, as the query results are always correct (sound and complete) with respect to the computed conjunctive query and the given dataset. Instead, a potential cause of imprecision lies in the translation of the keyword queries. In the evaluation, we thus put particular emphasis on this step.

4.2 Evaluation Results

We now report on the evaluation results, discussing first the overall effectiveness and efficiency, and then in more detail specific aspects of the individual steps of our search process. Due to space constraints, we present a much condensed form. A more detailed analysis with the complete results can be found in our Technical Report [8].

Overall Effectiveness and Efficiency. To measure the overall effectiveness, we have analyzed the ratio of tasks that have been successfully completed. To assess the efficiency, we measured the number of keyword queries that the user had to issue in order to complete the task. The results for these measures are shown in Figure 4. The results are aggregated over all queries (125 in total), grouped by the type of query that was needed to obtain the result. For the simple tasks (entity queries), the success rate was

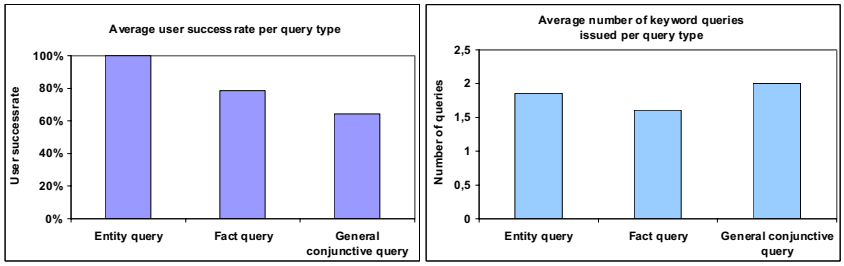

Fig. 4. Efficiency and Effectiveness of the Search

100%, the more complex tasks result in lower success rates: 79% for the fact queries and 64% for the general conjunctive queries. A typical reason why particular tasks were not completed was that the matching of the keywords against the available data was unsuccessful and no interpretations could be generated: The gap between the keywords and the underlying data was too large in certain cases. The more complex the queries (both in terms of structure and number of keywords), the larger was the effect (see discussion of keyword translation below).

Overall, 6 out of 14 participants were able to fulfill all tasks, 12 of the 14 were able to fulfill 60% or more. The other two users quickly gave up after the first or second query stating that they found the system too complicated. We find the overall success rate rather encouraging, considering that the participants used the system without detailed usage instructions and without knowing the schema of the underlying data.

For the efficiency, we see that on average the users needed to issue between 1.6 and 2 keyword queries to fulfill a task, depending on the query type. This number is surprisingly low, considering the structure and complexity of the generated queries and results. Expectedly, the value is larger for the more complex, general conjunctive queries (2 keyword queries per task) than for the simple types of queries.

We now discuss specific aspects of the three steps involved in our search process. In the questionnaire, we asked various questions related to the individual steps. The responses to these questions are shown in Figure 5.

Articulation of the information needs. The first question asked how difficult the users found it to express the information need in keywords. As expected, the users found it rather easy to do so, as all of them were familiar with keyword-based search interfaces.

Query interpretation by keyword translation. The next aspect we analyzed was the quality of the translation of the keyword queries to structured queries. First, we analyzed the robustness of the keyword matching: 88% of the keyword queries could be translated into structured queries. As mentioned above, the main reason why certain keyword matches failed was a gap between the keywords and the underlying data. As the users were not aware of the underlying schema at all, in some case no matching to the underlying data could be performed. However, after a more detailed analysis of the failures, we were able to improve the keyword matching algorithm after the user study.

Second, we measured the quality of the rankings of the possible interpretations. For this, we analyzed, which interpretation was selected by the user as the correct interpretation. We adopted a standard IR metric called Reciprocal Rank (RR) defined as

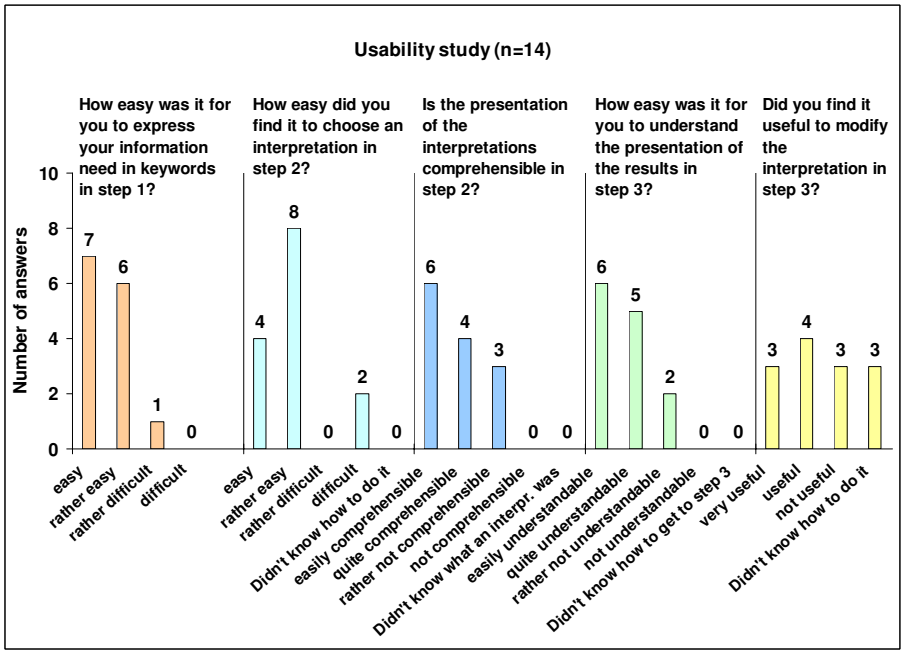

Fig. 5. Results of the usability study

$RR = \frac{1}{r}$, where r is the rank of the correct query. The mean RR for all queries was 0.84. For the successfully completed tasks, the users selected the top-ranked interpretation in 76% of the queries. The results indicate that the intended interpretation in most cases was ranked correctly, i.e. at first position.

Overall, the users found the representation of interpretations easily comprehendible, and it was easy for them to choose the right interpretation. Yet a few users had difficulties (see questions regarding step 2 in Figure 5), one reason being that in some cases the interpretations were so similar that the users could not easily tell the difference.

Result Presentation and Query Refinement. Finally, we analyzed the user satisfaction with the result presentation and query refinement. The majority of the users found the presentation of the results understandable. However, only seven users made use of the faceted search to refine a query. This corresponds to the question about how useful the modification was to the users: Seven participants found it *very useful* or *useful* to modify the interpretations, whereas three participants stated that they *did not know how to do it*. This suggests that it is useful, if the users know how to perform refinements. Unlike keyword search, faceted search is not a commonly used paradigm yet. As a consequence, effective use requires more detailed instructions, which were (deliberately) not given in our setting. Interestingly, the use of the faceted search was particularly effective for the more complex tasks. On average, 29.6% of the successfully completed tasks involved a refinement. For the most complex tasks involving general conjunctive queries, 38.9% of the successfully completed tasks involved refinements. We thus have

reasons to believe that the overall success rate would have been much higher, if all users had known how to effectively utilize the faceted search.

Response Times. The evaluation of the system performance in terms of response times was not a primary goal of our study, as this aspect already had already been extensively analyzed in our prior work [5]: We showed that the query translation and query processing can be handled in near real-time even with considerably larger data sets. Still, for completeness we measured the time to translate keyword queries as well as the time for query answering. The keyword translation was performed on average in 132 ms, while the query answering (evaluation of the conjunctive query) on average took 31 ms.

5 Related Work

Our research relates to work from the three major areas, namely (1) search in wikis, (2) semantic search based on keyword translation, (3) faceted search.

Search in wikis. For wikis as knowledge sharing systems it is an important question how the knowledge in the wiki can be accessed, searched, and queried. MediaWiki, like most wiki systems, comes with a standard full-text search, which supports users to find pages in the wiki based on keyword matches. However, no structural information is exploited and only pages are regarded as search results. Also other semantic wiki systems, like AceWiki[3] or IkeWiki [4], which are powerful tools for editing and presenting structured data, do not take advantage of the structured data for search other than the standard full-text keyword search. Making the knowledge of Wikipedia accessible for structured queries is the goal of dbpedia[5], which extracts structured data from Wikipedia (and other datasets) and provides public access to it. However, users still need to know the syntax of formal query languages to search its content, which is not suitable for lay end users even though a graphical query builder OpenLink QBE [6] supporting the construction of queries is available.

Semantic search based on keyword translation. [3] provides a review of different semantic search tools and focuses on different modes of user interaction. Compared with other modes of interaction (form-based, view-based, or natural language), the advantages of keyword based lie in its simplicity and the familiarity most users already have with it. The problem of keyword queries on structured data has been studied from two different directions: 1) computing answers directly through exploration of substructures on the data graph [6,7] and 2) computing queries through exploration of a query space [5]. It has been shown in [5] that keyword translation operates on a much smaller query space, and is thus efficient. Besides, the structured queries presented to the user help in understanding the data (answer) and allow for more precise query refinement. We follow the second direction to keyword search and adapt it to wiki-based environments. [9] gives an analysis of existing semantic search systems, identifies different types of

[3] `http://attempto.ifi.uzh.ch/acewiki`
[4] `http://ikewiki.salzburgresearch.at`
[5] `http://dbpedia.org`
[6] `http://dbpedia.openlinksw.com:8890/isparql`

common features and also points out the lack of both the evaluation of search algorithms on publicly available real world datasets and the evaluation of user interfaces for semantic search. We addressed the first in our previous work [5] and the latter with the user evaluation presented in Section 4.

Faceted Search. Faceted Search (or Faceted Browsing) is increasingly used in search applications, and many websites already feature some sort of faceted search to improve the precision of their website search results. A crucial aspect of faceted search is the design of a user interface, which offers these capabilities in an intuitive way. This has been studied by [10,11] and applied in systems like Flamenco[7], Exhibit[8] or Parallax[9]. In a Semantic MediaWiki context, this paradigm has been applied by Semantic Drill Down[10] for browsing from top to bottom along the wiki's categories. Another cornerstone of faceted search is the question what is actually used as facets and if they are hierarchical or multidimensional, which obviously depends on the data corpus and its structure. Flamenco and Exhibit require a predefined set of properties, which each data item has to have and then allows browsing along the values of these properties. We use parts of the schema as facets, and therefore keep a domain independent and multidimensional faceted search. By multidimensional, we mean that vertical browsing is possible too. Although we precompute the schema, which is an indirect preparation of the facets, our approach determines the facets on the fly and provides those leading to non empty result for the each interpretation.

6 Conclusions and Future Work

We have presented an approach to search in semantic wikis, in which search is considered as a process with active user involvement: Information needs are articulated using keyword queries, which are translated into possible interpretation in terms of conjunctive queries. Users can select the interpretation and further refine it using faceted search. We have implemented this approach in an extension to Semantic MediaWiki and evaluated it in one of the most prominent installation of SMW, *semanticweb.org*.

The advantage of the approach lies in the combination of the expressiveness of structured query languages with the simplicity of keyword search and faceted search: While the user does not need to know the schema and structure of the underlying data, complex information needs can be answered that with prior technology would require either manual gathering of information (potentially distributed over a large number of pages), or the use of the formal syntax of structured query languages.

The results of our user study show that the search process is indeed adequate for end user: The participants of the study were able to use the search interface immediately without detailed instructions. Expressing the information needs in keywords, understanding and selecting interpretations in terms of conjunctive queries, and understanding the results caused little or no difficulties. However, the faceted search, while

[7] `http://flamenco.berkeley.edu`
[8] `http://simile.mit.edu/exhibit`
[9] `http://mqlx.com/~david/parallax/`
[10] `http://semantic-mediawiki.org/wiki/Help:SMW_extensions#Semantic_Drilldown`

extremely useful for refining queries to match the exact information need, was found to be too difficult by some users. Overall, it seems that people are used to the simplicity of the Google search process, and even if added value is provided, it is hard to depart from that. In this regard, there are several possibilities how the process can be simplified in certain cases: The second step in many cases already fulfills the information need: Namely, in the cases of queries for single entities and facts, the result snippets shown for the possible interpretations already contain the complete answers. Further, as our analysis of the query ranking shows, the top-ranked interpretation matches the intended user information need in most cases. Therefore, it seems reasonable to (optionally) skip the second step and show alternative interpretations only on request.

There are several further directions for future work. Currently, our search operates on the structured content of the wiki available in OWL. In this regard, we plan to extend our work to support a combination of keyword search over structured *and* unstructured content, building on our existing work on hybrid search [12]. Further, as Semantic MediaWiki is extended to support larger fragments of OWL, we intend to investigate how this can be supported and exploited for semantic search. Finally, we plan to do studies with installations of our search system in semantic wikis in other domains, where users have a less technical background.

Acknowledgements. Research reported in this paper was partially supported by the EU in the IST projects NeOn (IST-2006-027595), `http://www.neon-project.org/` and X-Media (IST-2006-026978), `www.x-media-project.org`

References

1. Guha, R., McCool, R., Miller, E.: Semantic search. In: WWW 2003: Proceedings of the 12th international conference on World Wide Web, pp. 700–709. ACM, New York (2003)
2. Mangold, C.: A survey and classification of semantic search approaches. International Journal of Metadata, Semantics and Ontologies 2(1), 23–34 (2007)
3. Uren, V.S., Lei, Y., Lopez, V., Liu, H., Motta, E., Giordanino, M.: The usability of semantic search tools: a review. Knowledge Eng. Review 22(4), 361–377 (2007)
4. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantics 5(4), 251–261 (2007)
5. Tran, D.T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query graph candidates for efficient keyword search on rdf. In: Proceedings of the 25th International Conference on Data Engineering (ICDE 2009), Shanghai, China (May 2009)
6. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: Chan, C.Y., Ooi, B.C., Zhou, A. (eds.) Proc. of the 2007 SIGMOD Int. Conf. on Management of data, pp. 305–316. ACM, New York (2007)
7. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: Proc. of the 31st Int. Conf. on Very Large Data Bases (VLDB), pp. 505–516. ACM, New York (2005)
8. Haase, P., Herzig, D., Musen, M., Tran, T.: Semantic wiki search. Technical report, University of Karlsruhe (2008), `http://www.aifb.uni-karlsruhe.de/WBS/pha/publications/semwikisearch.pdf`

9. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: An analysis of search-based user interaction on the semantic web. Technical report, CWI, Amsterdam, The Netherlands (2007)
10. Hearst, M., Elliott, A., English, J., Sinha, R., Swearingen, K., Yee, K.P.: Finding the flow in web site search. Communications of the ACM 45(9), 42–49 (2002)
11. Hearst, M.: Design recommendations for hierarchical faceted search interfaces. In: ACM SIGIR Workshop on Faceted Search (August 2006)
12. Wang, H., Tran, T., Liu, C.: Ce2: towards a large scale hybrid search engine with integrated ranking support. In: CIKM, pp. 1323–1324. ACM, New York (2008)